Instituto Federal de Educação, Ciência e Tecnologia do Amazonas - IFAM

Campus Manaus Zona Leste

Disciplina: Sistemas Operacionais

Professor: Albert França Josuá Costa

Aluna: Brenda Cristina Dourado Moura

RELATÓRIO DA LISTA 02 - ESCALONADORES

Relatório solicitado pelo professor Albert França Josuá Costa para obtenção da nota parcial da disciplina de Sistemas Operacionais, a qual compõe o quarto semestre do curso de Engenharia de Software.

MANAUS - AM

11/2021

**Sumário**

**Introdução 1**

**Recursos utilizados 2**

**Atividade desenvolvida 3**

**Resultados e conclusão 6**

**Referências 7**

INTRODUÇÃO

Sistemas Operacionais (SO) são programas que agem na camada intermediária entre o computador e o hardware, realizando duas funções básicas. A primeira função é servir como máquina virtual, abstraindo assim detalhes de programação de hardware. Outra função bastante importante do SO é gerenciar processos (programas em execução). Um subtópico importante relacionado aos sistemas operacionais é relacionado aos processos e seus escalonadores, é importante entender que os escalonadores são os responsáveis por definir qual será o próximo processo que utilizará os recursos do processador, Existem quatro tipos básicos de escalonadores: O primeiro é o escalonador do tipo FIFO, como é sugerido por seu nome, a ordem de utilização do processador é definida pela ordem de chegada dos processos ao escalonador, este tipo possui como maior desvantagem o fato de que processos de menor importância e maior duração prejudiquem o funcionamento do processo, caso sejam adicionados primeiro ao processador. O próximo tipo é o escalonador por prioridade, nesse tipo o importante é a prioridade atribuída ao processo, processos com maior prioridade são executados primeiro. O terceiro é o Shortest Job First, nesse escalonador o processo possui um tempo e o escalonador é responsável por ordenar os processos a partir do menor tempo, ou seja, processos com o menor tempo atribuído será executado primeiro. O último é conhecido por Circular Round Robin, nesse tipo de escalonador o processo, além de possuir o tempo total de execução, também possui um atributo chamado de Quantum, esse atributo define o tempo-limite de utilização continua do processador, ao finalizar esse tempo o processo, caso ainda não tenha finalizado o tempo de execução, volta à fila de aptos. Para o bom funcionamento e implementação das filas de uso do processador, é importante compreender os estados que um processo pode assumir, sendo eles: criado, apto, executando, bloqueado e destruído. Este trabalho propõe construir uma simulação do funcionamento dos escalonadores do tipo FIFO, por Prioridade, Shortest Job First e Circular Round Robin. Esta abordagem de aprendizado prático visa estimular o aprendizado acerca do real funcionamento dos escalonadores.

RECURSOS UTILIZADOS

* **Linguagem de programação:**

Para a implementação de processos e dos escalonadores dos tipos FIFO, por Prioridade, Shortest Job First e Circular Round Robin foi necessário a utilização da linguagem de programação C, criada em 1970 por Dennis Ritchie, essa linguagem é de alto nível, ou seja, seus comandos são criados o mais próximo possível da linguagem humana, no caso do C, os comandos são em inglês.

* **Ferramentas:**

Para o desenvolvimento dos códigos foi utilizada a IDE Code Blocks com o compilador GNU GCC.

* **Estruturas:**

Para a construção da solução foram utlizados ponteiros, que consistem em variáveis especiais que armazenam endereços, além disso, também foram utilizadas structs (junção de uma ou várias variáveis com compõem o mesmo tipo de dado).

ATIVIDADE DESENVOLVIDA

* **Objetivo do programa:** 
  + Criar os algoritmos de escalonamento do tipo FIFO, por Prioridade, Shortest Job First e Circular Round Robin, simulando a execução dos processos por meio da escrita da string “Processo + identificador\_do\_processo” em um arquivo de texto a cada interação do programa, com o arquivo de texto não sendo sobrescrito.
* **Resumo das atividades desenvolvidas – Geral:**
  + **Structs criadas:**

//Estrutura do nó

|  |
| --- |
| Struct no{ |
|  | int identificador; |
|  | int quantum; |
|  | int tempo; |
|  | int prioridade; |
|  | int estado; |
|  |  |
|  | //Arvore |
|  | struct no \*pai; |
|  | struct no \*primFilho; |
|  | struct no \*proxIrmao; |
|  |  |
|  | //Lista encadeada |
|  | struct no \*prox; |
|  | struct no \*ant; |
|  | }; |

//Identidade

|  |
| --- |
| struct registrador{ |
| struct no \*raiz; |
|  |
| struct no \*cabeca; |
| struct no \*cauda; |
| }; |

Obs.: para a elaboração do trabalho foi criado uma estrutura híbrida, juntando conceitos de fila e árvore, dessa forma não foi necessário a criação de uma struct auxiliar com informações separadas sobre a lista encadeada.

* + **Funções criadas:**
    - tornarApto() – Criada para transformar em aptos os processos que estão na fila de criados.
    - buscarProcesso () – Criada para buscar um processo dentro da árvore.
    - inicializa() – Criada para inicializar a identidade da estrutura principal.
    - novoProcesso() – Criada para criar um novo nó e preencher suas informações base.
    - inserirProcesso() – Criada para inserir um novo nó na árvore já, organizando as informações referente à fila.
    - criarProcessoInicial() – Criada para criar o processo inicial da árvore.
    - quantidadeProcessos() – Criada para retornar a quantidade total de processos adicionados na árvore.
    - imprimirProcessos() – Criada para imprimir as informações de processos adicionados na árvore.
    - tempoAtual() – Criada para retornar o tempo atual do Sistema.
    - escreveDocumento() – Criada para acrescentar um texto (“Processo + identificador\_do\_processo”) em um arquivo de texto.
    - executando() – Criada para executar a simulação de andamento de processos no ciclo do processador.
    - main() – Função principal do programa, responsável por verificar informações de leitura de processos,verificação da opção do menu que foi escolhida pelo usuário e outras.
  + **Informações importantes:**
    - O processo inicial possui como valor padrão o número zero.
    - A leitura de um novo processo é realizada por meio de uma string com formato “PX-PY”, onde P é sempre a letra P maiúscula ou minúscula, X é o número do pai do processo e Y é o número do processo. Para que um processo seja criado é necessário que exista um pai pré-existente e que a chave do processo (Y) ainda não tenha sido cadastrado no programa, caso uma dessas situações ocorra, o programa informará ao usuário e solicitará que a leitura do processo ocorra novamente.
  + **Fluxo principal do programa:**

1. Rodar o programa.
2. Selecionar a opção de criar processos.
3. Informar a quantidade total de processos a serem lidos pelo programa.
4. Digitar as informações dos n processos.
5. Selecionar a opção de iniciar a simulação.
6. Solicitar que o programa imprima o diagrama de processos.
7. Selecionar a opção de contagem dos n processos existentes.
8. Encerrar o programa.

* **Resumo das atividades desenvolvidas - Escalonadores:**
  + **Tipo FIFO:** para esse tipo de escalonador os processos utilizaram parte referente à fila na estrutura híbrida, sendo adicionados à fila na mesma ordem que foram criados pelo usuário.
  + **Tipo por Prioridade:** o diferencial desse escalonador é a ordenação por prioridade no momento de adicionar na lista encadeada. Processos com a maior prioridade eram adicionados mais para o início da lista.
  + **Tipo Shortest Job First:** como o tipo de escalonador anterior, foi necessário que a lista fosse ordenada, nesse escalonador, os processos foram ordenados pelo tempo, ou seja, processo que possuíam o menor tempo total para execução eram adicionados mais para o início da lista.
  + **Circular Round Robin:** (a fazer)

RESULTADOS E CONCLUSÃO

REFERÊNCIAS

https://www.ti-enxame.com/pt/c/como-dividir-uma-string-em-2-strings-em-c/968225763/

https://www.ti-enxame.com/pt/c/como-remover-o-caractere-em-um-vazio-indice-de-uma-string-em-c/971615844/

https://labdegaragem.com/forum/topics/converter-char-para-int

<https://www.ti-enxame.com/pt/c/como-voce-limpa-tela-do-console-em-c/968047218/>

<http://linguagemc.com.br/exibindo-data-e-hora-com-time-h/>

http://linguagemc.com.br/arquivos-em-c-categoria-usando-arquivos/

<http://wiki.icmc.usp.br/images/8/82/Manipulacao_arquivos.pdf>

<https://www.inf.ufpr.br/wagner/so/processos+threads.2pp.pdf>

<https://www.gsigma.ufsc.br/~popov/aulas/so1/cap8so.html>

<http://proedu.rnp.br/bitstream/handle/123456789/711/Sistemas_Operacionais_web.pdf?sequence=3&isAllowed=y>

<https://www.seduc.ce.gov.br/wp-content/uploads/sites/37/2011/10/redes_de_computadores_sistemas_operacionais.pdf>

<http://www.facom.ufu.br/~gustavo/ED1/Apostila_Linguagem_C>

https://www.inf.pucrs.br/~pinho/LaproI/Structs/Structs.htm

<https://www.treinaweb.com.br/blog/ponteiros-em-c-uma-abordagem-basica-e-inicial>

https://www.inf.pucrs.br/~pinho/LaproI/Structs/Structs.htm